PLDshell Plus™/PLDasm™ User's Guide Supplement

V2.6

Order Number: 610959-001

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

376, Above, ActionMedia, BITBUS, Code Builder, DeskWare, Digital Studio, DVI, EtherExpress, ETOX, FaxBACK Grand Challenge, i, i287, i386, i387, i486, i487, i750, i860, i960, ICE, iLBX, Inboard, Intel, Intel287, Intel386, Intel387, Intel486, Intel487, intel inside, Intellec, iPSC, iRMX, iSBC, iSBX, iWARP, LANPrint, LANSelect, LANShell, LANSight, LANSpace, LANSpool, MAPNET, Matched, MCS, Media Mail, NetPort, NetSentry, OpenNET, PRO750, ProLolver, READY-LAN, Reference Point, RMX/80, SatisFAXtion, SnapIn 386, Storage Broker, SugarCube, The Computer Inside., TokenExpress, Visual Edge, WYPIWYF.

MDS is an ordering code only and is not used as a product name or trademark. MDS is a registered trademark of Mohawk Data Sciences Corporation.

MULTIBUS is a patented Intel bus.

CHMOS and HMOS are patented precesses of Intel Corp.

Intel Corporation and Intel's FASTPATH are not affiliated with Kinetics, a division of Excelan, Inc. or its FASTPATH trademark of products.

PLDshell Plus and PLDasm are also trademarks of Intel Corp.

PLDshell Plus has patents pending.

PAL and PALASM are registered trademarks of Advanced Micro Devices, Inc.

GAL is a registered trademark of Lattice Semiconductor, Inc.

The installation program used to install PLDshell Plus, *INSTALL*, is based on licensed software provided by Knowledge Dynamics Corp., Highway Contract 4 Box 185-H, Canyon Lake Texas 78133-3508 (USA), 1-512-964-3994, *INSTALL* is Copyright © 1987-1990 by Knowledge Dynamics Corp., which reserves all copyright protection wordwide. *INSTALL* is provided to you for the exclusive purpose of installing PLDshell Plus. Intel has made modifications to the software as provided by Knowledge Dynamics Corp., and thus the performance and behavior of the *INSTALL* program shipped with PLDshell Plus may not represent the performance and behavior of *INSTALL* as shipped by Knowledge Dynamics Corp. Intel is exclusively responsible for the support of PLDshell Plus, including support during the installation phase. In no event will Knowledge Dynamics Corp. be able to provide any technical support for PLDshell Plus.

Table of Contents

Chapter 1 — Introduction

This document is a supplement to version 2.1 of the *PLDshell Plus/PLDasm User's Guide*. To fully use version 2.6 of PLDshell Plus, you must be familiar with PLDshell Plus/PLDasm.

New Features of PLDshell Plus/PLDasm

This document describes the new features of PLDshell Plus and PLDasm 2.6. Version 2.6 provides preliminary support for Intel FPGA devices. The new features include:

- Extended Memory support
- Semi-automatic design merger
- SRAM configuration syntax
- Compare operation syntax
- Syntax for:
  - Buried macrocells
  - Delayed Clock
  - 3.3V/5V outputs
  - Open Drain outputs
  - TTL or CMOS inputs
- Vector Notation
- Functional simulation to support FPGA
- Access to all internal signals in simulation
- Added language notation for grouping/modular design
- Integrated mouse support
- Improved Help facility

Environment

- MS-DOS 3.1 or later
- IBM PC/AT, 386-AT, or true compatible (minimum 80386)
- VGA monitor (to view waveforms)
- Minumum of 2 MB of extended (XMS) memory
- Minimum of 5 MB of hard disk space

- 1.44 MB (3.5 inch) floppy diskette drive

Using Extended (XMS) Memory

Using Extended (XMS) Memory with PLDshell Plus requires that you use an extended memory manager. In general, the following programs are compatible with PLDshell Plus:

HIMEM.SYS
RAMDRIVE.SYS
SMARTDRV.SYS
Qualitas 386MAX

The appropriate driver programs should be included in your CONFIG.SYS file. At a minimum, an extended memory manager is *required*. Refer to your MS-DOS or Qualitas documentation for complete information on these programs.

The program HIMEM.SYS is not compatible with VDISK.SYS. If both programs are installed, the result will be no extended memory available for other application programs. Instead, use the RAMDRIVE.SYS program.

NOTE

PLDshell Plus may not be compatible with some versions of disk caching programs. If you experience a problem installing PLDshell Plus, disable the disk cache program before installing PLDshell Plus.

Chapter 2 — User Interface

New Features

The PLDshell 2.6 user interface includes the following additional features:

- Mouse support: select menus, options, fields, and open, close, move and resize windows with the mouse
- Status screen added as first-line Help
- Resource/Device Estimator
- New Merge Utility

Mouse Support

All PLDshell window items can be selected using the mouse. The mouse-support features allow you to:

- Pull down menus
- Select fields, filenames, and options
- Manipulate windows.

"Selecting" an item with the mouse means placing the cursor on the item or in a field and pressing the left mouse button. This is also known as "clicking" on a menu item or field.

Fields that display lists can be displayed by double clicking in the field.

Close Boxes

Figure 2-1 shows an example list window with a **Close** box. List and view windows without **Cancel** or **Accept** buttons have **Close** boxes. This is the same as pressing the <ESC> key.

Resizable/Moveable View Windows

Figure 2-2 shows a view window that can be resized or moved with the mouse. Place the mouse cursor on sizing button at the bottom right corner of the window and use click and drag to resize the window. To move a view or list window, place the mouse cursor on any window border, except a scroll bar (if any), and click and drag the window.

NOTE
Any corner without a close box can be resized.

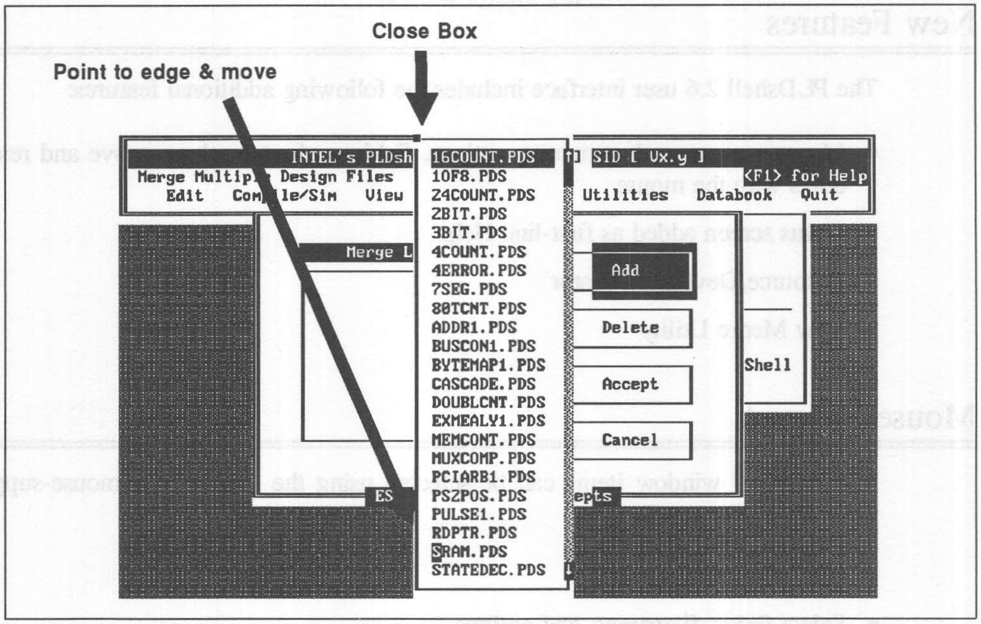


Figure 2-1. Window Close Boxes

Scroll Bars

View windows have scroll bars at the right and bottom borders (Figure 2-2). To scroll

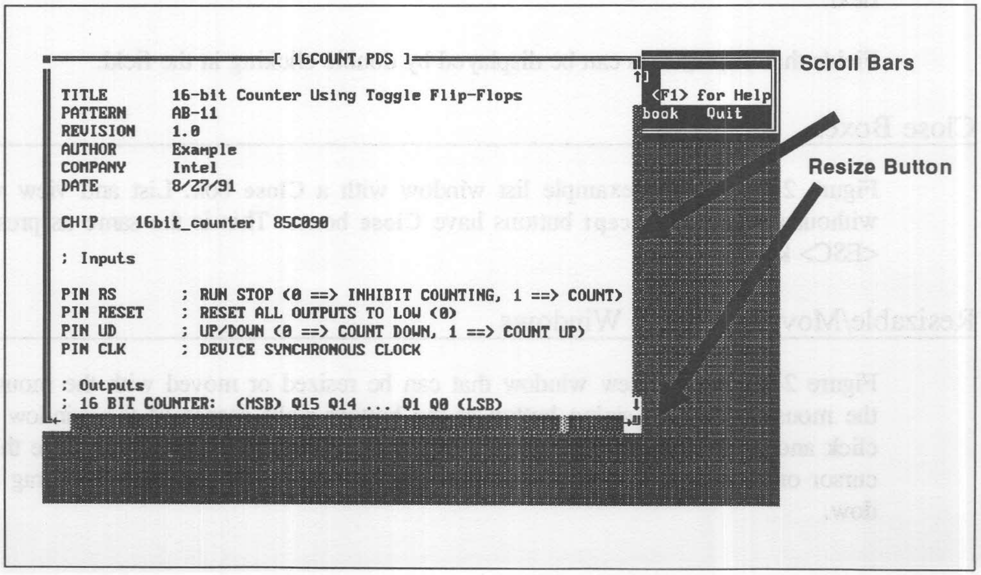


Figure 2-2. Resizable View Window

through a file, place the mouse cursor in the grayed areas of the scroll bar and click the left mouse button. You can also click on the arrows at each end of a scroll bar.

Help

Figure 2-3 shows an example of the Help/Status screen. This is a new screen that provides a first level of help.

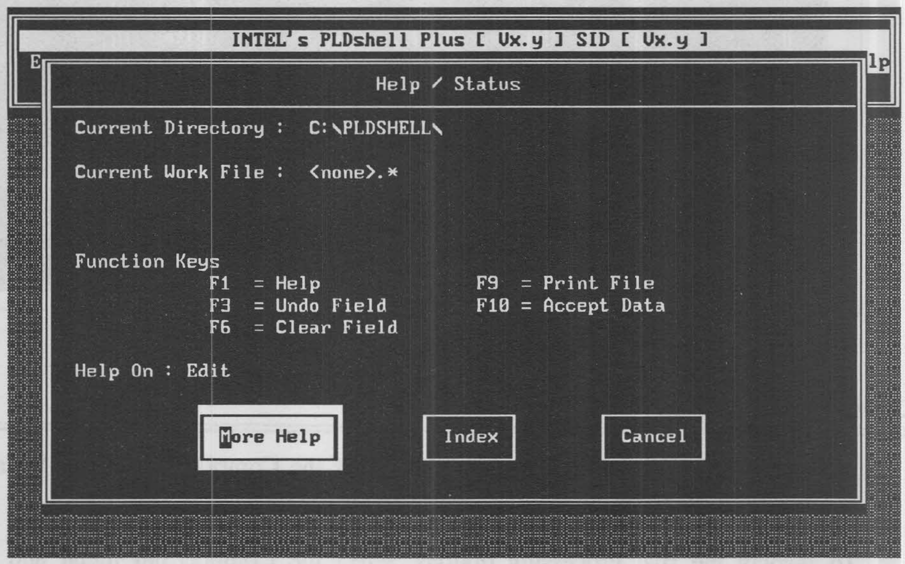


Figure 2-3. Example Help/Status Screen

The new Help/Status window displays:

- The current working directory
- Current work file
- Function Key Summary
- Current Topic

There are three buttons:

More Help — Displays more Help information on the current topic

Index — Displays a logically ordered list of Help topics (see Figure 2-4).

Cancel — Cancels the help facility and returns to the current menu or submenu.

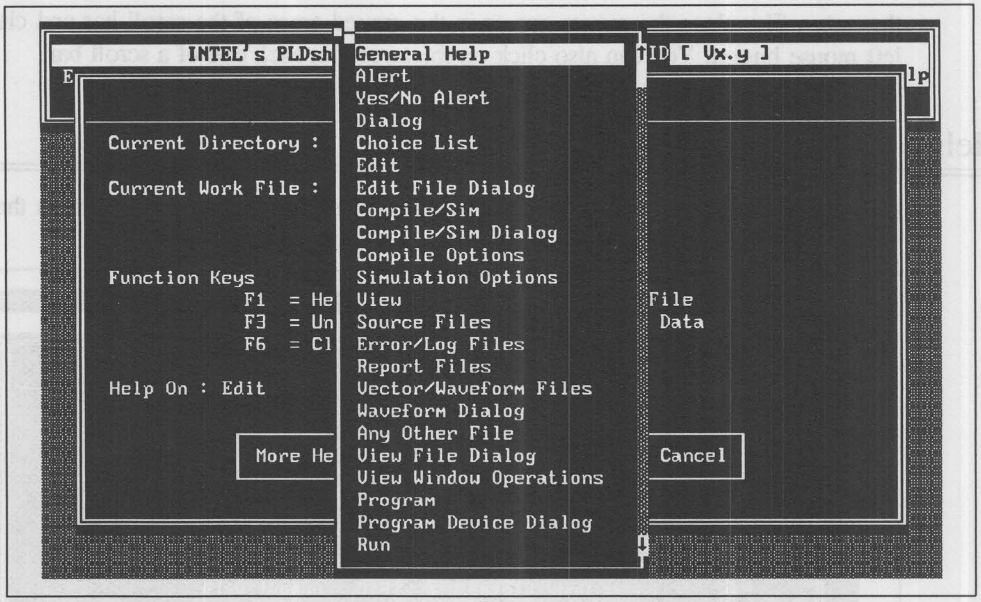


Figure 2-4. Help Index Screen

Accessing the Estimator

Figure 2-5 shows the two Estimator options of the Compile/Sim menu: Estimate Only and Estimate Then Simulate.

To display the five Processing options, select the Compile/Sim menu and move the cursor to the Processing field and press the <SPACE> key. You can also double click in the Processing field to display the list of processing options. The two additional options are:

> **Estimate Only** — Parses in the file and minimizes the equations (if selected). The Estimator is then run to make a rough estimate as to which devices are likely to fit the design.
>
> **Estimate Then Simulate** — Parses in the file and minimizes the equations (if selected). The Estimator is then run to make a rough estimate as to which devices are likely to fit the design. After estimation, the simulator is run (if a simulation section is present).

Select the desired processing mode by clicking on the that mode.

You can also double-click in the Input Filename field to display a list of .PDS files.

Figure 2-5. Estimator Access

Auto Display Report

Figure 2-6 shows the new Auto Display Report option on the Compile Options sub-menu. This option automatically displays the compiler report file (.RPT) or Estimator file (.EST) upon completion of Compiler or Estimator operation, respectively. The report file will be displayed only if the Report File setting is also "Yes."

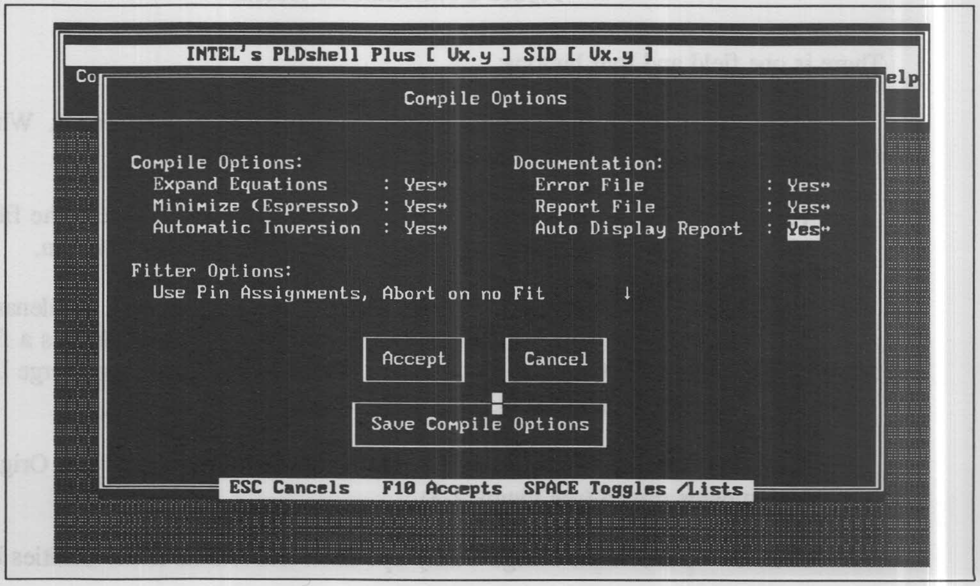


Figure 2-6. Auto Display Report Compiler Option

Merge

PLDshell V2.6 includes a Merge option. This feature allows merging up to eight files into a single file for compiling, simulating, and/or estimating. Figure 2-7 shows the initial Merge screen. A merging example is described in detail in Chapter 4, "Using Merge."

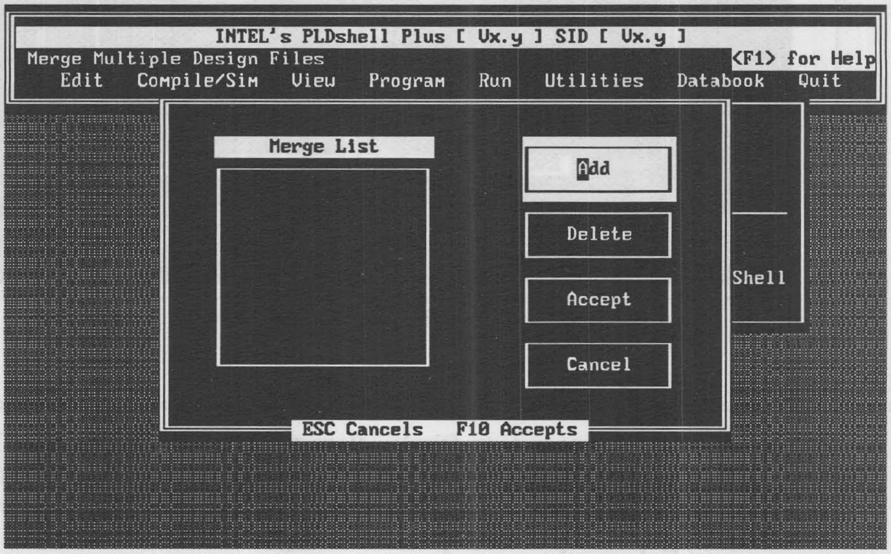


Figure 2-7. Estimator Access

There is one field and four buttons:

Merge List — Displays a list of up to eight files to be merged. When first invoked, the list is empty. The minimum number of files is two.

Add — Adds a filename to the Merge List. Click on a filename in the file list to add a file to the Merge List. Only one file is added with each operation.

Delete — Deletes a filename from the Merge List. Displays a list of filenames that matches the Merge List. Clicking on a filename in the file list deletes a file from the Merge List. Deletes one file at a time. If only one file is in the Merge List, that file is deleted and the file delete window closes.

Accept —Accepts the files in the Merge List and displays the Use Original Pin Assignments screen (see Figure 2-8).

Cancel — Cancels the Merge Utility operation and returns to the Utilities Menu.

Use Original Pin Assignments Screen

Figure 2-8 shows the Use Original Pin Assignments screen.

Figure 2-8. Use Original Pin Assignments

This screen displays the files in the Merge List (up to eight files). Each file is preceded by a pair of brackets.

Initially, the space between the brackets is blank. If you choose to use the original pin assignments for one or more files, click on the brackets and an "X" will appear. Click again to remove the "X." The "X" indicates that the original pin assignments for a file will be used during the Merge operation.

Accept — Accepts the contents of this screen and displays the Resolve I/O Signals Names/Types screen (see Figure 2-9).

Cancel — Cancels the current operation and returns to the initial Merge screen with the filenames in the Merge List.

Resolve I/O Signals Names/Types Screen

Figure 2-9 shows the Resolve I/O Signals Names/Types screen.

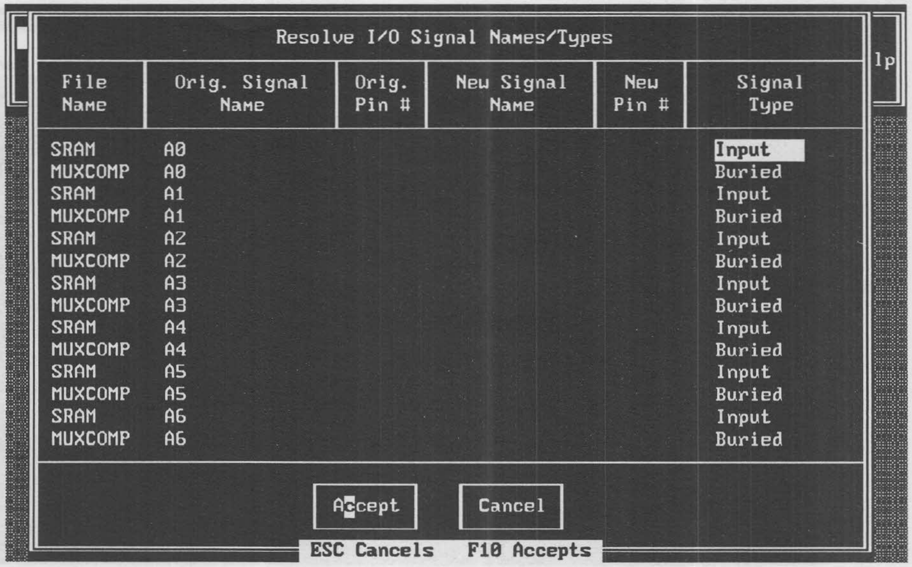


Figure 2-9. Resolve I/O Signals Names/Types Screen

This screen contains six columns. The three columns on the left cannot be edited; the three right-hand columns can be edited.

Source File — Filename of the source file for a particular signal.

Orig. Signal Name — Name of the signal in the source file.

Orig. Pin # — Pin number in the source file. If you chose not to use the original pin assignments, this field will be blank.

New Signal Name — Signal name in the merged design. This is determined by the user if a new name is desired.

New Pin # — The pin number in the merged design. The original pin numbers of a file, if selected, will be displayed. In no number is displayed, you can enter a pin number if desired. Alternatively, this field can be left blank and pin numbers can be assigned by the compiler.

Signal Type — Input, Output, or Buried. Inputs cannot be changed. You can toggle outputs between Output or Buried signal type via the <Space> bar or mouse click.

You can use the arrow keys or PgDn and PgUp to scroll through the list.

Accept — Accepts the contents of the current screen and displays the **Merge Design Files** screen (see Figure 2-10).

Cancel — Cancels the current operation and returns to the initial Merge screen, with the filenames in the Merge List.

Merge Design Files Screen

Figure 2-10 shows the Merge Design Files screen.

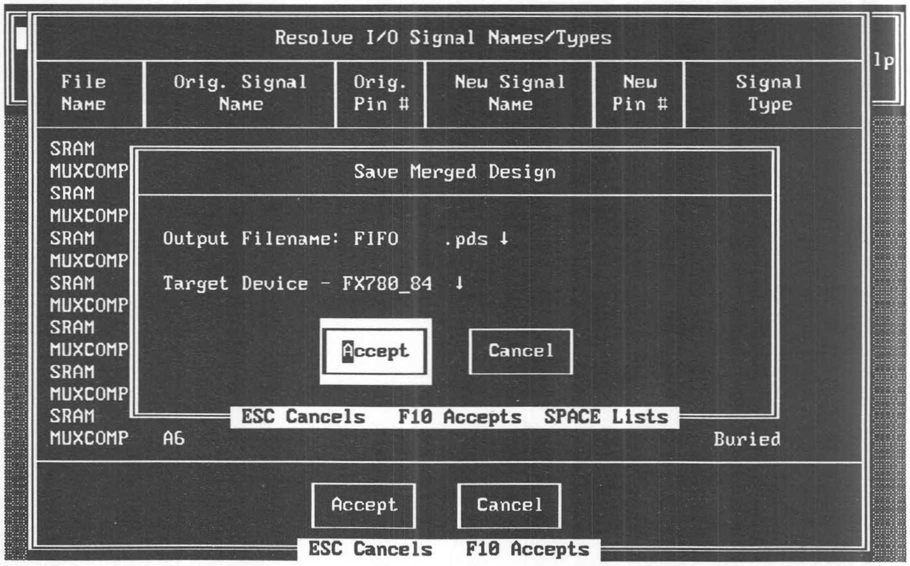


Figure 2-10. Merge Design Files Screen

This screen has two fields and two buttons.

Output Filename — This is the name of the merged PDS file. The extension .PDS cannot be changed. Enter the name of the merged file in this field. You can press the <SPACE> key or double click in this field to display a list of PDS files in the current directory.

Target Device — This is the target Intel device for the merge operation. Press the <SPACE> key or double click in this field to display a list of Intel FPGA and µPLD devices. Select the target device using the arrow keys and <Enter> or click with the mouse.

Accept — Accepts the contents of the two fields and creates the merged .PDS source file. When the file has been successful created, you will be prompted with an "OK" status screen.

Cancel — Cancels the current operation and returns to the Resolve I/O Signals Names/Types screen.

Chapter 3 — PLDasm Language Extensions

This section describes the language extensions to PLDshell Plus for supporting FPGA devices. The new features are:

- Hardware Compare (identity compare term)
- Signal Grouping
- 3.3V/5V/Open Drain outputs
- TTL/CMOS input levels
- Hierarchy
- Merging/modular design support
- SRAM
- Access to all internal signals during simulation
- Vector notation in design section
- Delayed Clock
- Buried Macrocells
- User Electronic Signature (UES) bits
- New I/O, feedback macrocell combinations

Hardware Compare

Figure 3-1 shows an example of the hardware compare. The hardware compare is an identity compare term that can feed the sum of products (SOP) expression of one of the outputs in a block. The compare term is not a sum of products function, and it will not be minimized by the compiler. A compare extension (.CMP) and double equal sign (==) have been added to the language to handle this feature.

The result of the comparison (cntl1 vs. addr1 and cntl2 vs. addr2) would be OR'd into the rest of the SOP expression for a macrocell. If a compare is the only function needed, then the first line could be omitted or set OUT = GND.

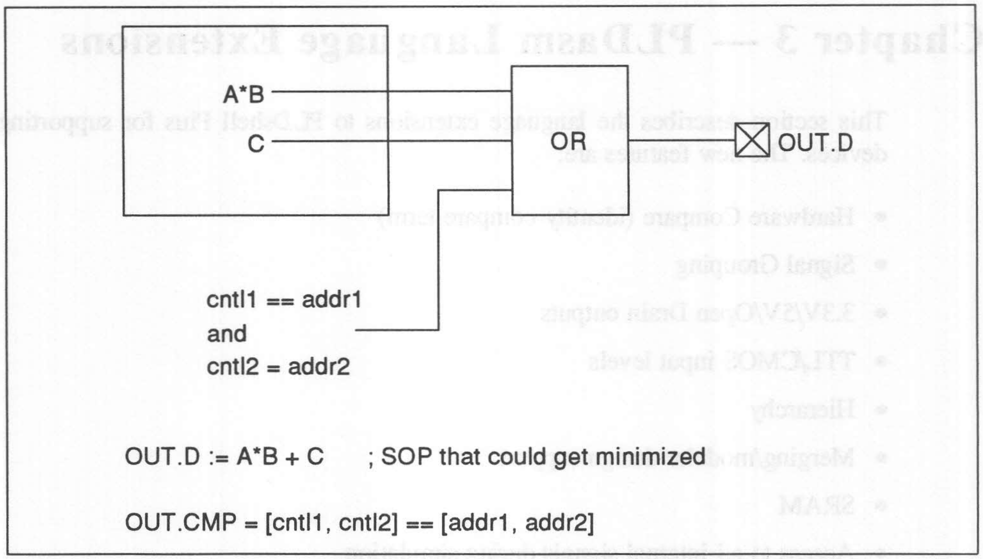


Figure 3-1. Hardware Compare Example

Signal Grouping

Figure 3-2 shows an example of the Signal Grouping feature, which allows signals to be grouped on adjacent pins if possible.

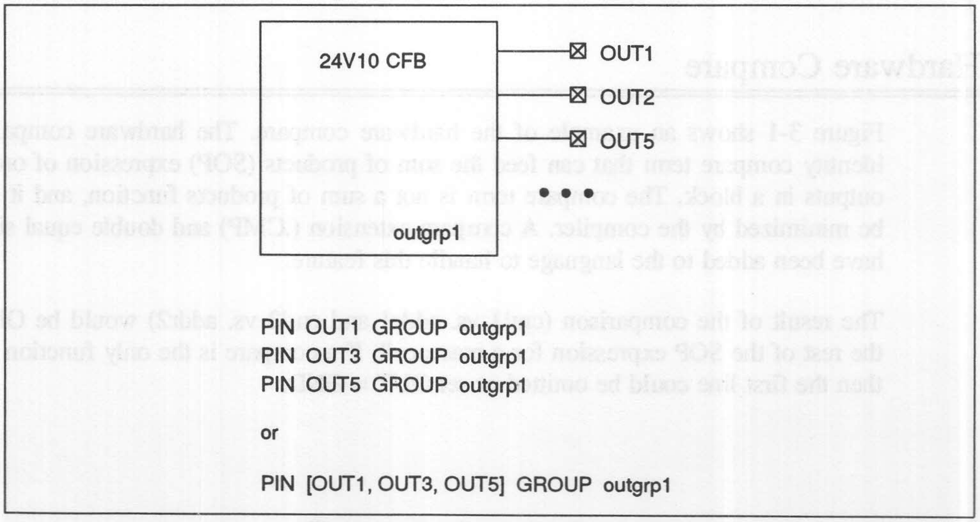


Figure 3-2. Grouping Example

Grouping should be specified in the pin declaration with the addition of a GROUP keyword, to be followed by a group name.

The statements in the example would mean that outputs OUT1, OUT2, and OUT3 are part of the group "outgrp1" and must be adjacent pins, but not necessarily in the same CFB.

3.3V/5V/Open Drain/Low Power

Keywords have been added to configure outputs and inputs to match application requirements. The following sections describe these keywords and show examples of their use.

Outputs

The specification of 3.3-volt versus 5-volt and standard versus Open Drain for output signals is handled in the PIN declaration keywords. New keywords have been added (3VOLT, 5VOLT, and OPEN_DRAIN) to specify 3.3-volt, 5-volt, and open drain outputs. The 3VOLT and 5VOLT keywords do not physically change the output levels. They allow the software to place these outputs in the same CFB. The VCCO pin on the CFBs are then tied to 3.3-volts or 5-volts.

The first line in Figure 3-3 defines Pin 12 as OUT1 and a 3.3-volt output.

The fourth line in Figure 3-3 is in the options section defining most or all outputs as 3.3-volts. This would mean that ALL outputs are 3.3-volt, unless explicitly changed on a pin declaration to be 5-volt (5VOLT). The software will group like voltage signals to the same CFB. The second line in the figure is a 5-volt output.

The OPEN_DRAIN keyword physically changes an output by disabling the internal pullup. An external pullup can then be used. The second line in Figure 3-3 shows an open drain example.

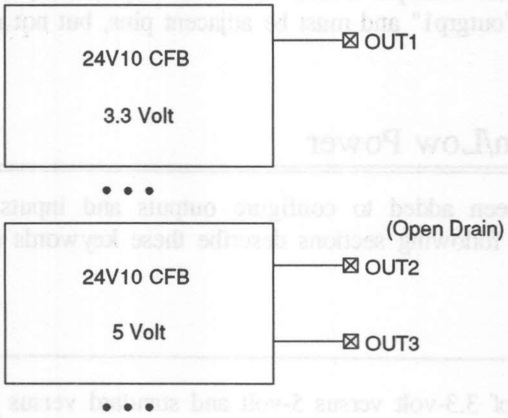
Inputs

Two keywords allow inputs and I/O pins used as inputs to be configured for CMOS or TTL levels. "CMOS_LEVEL" is used for 5V CMOS inputs. "TTL_LEVEL" is used for 3V CMOS and TTL input levels. This allows designers to optimize applications for t_{SB} (standby current) or leakage current in some situations. For example,

```
PIN  13  IN1  5VOLT  CMOS_LEVEL
or
PIN  18  IN2  3VOLT  TTL_LEVEL
```

The options section can be used to change the default input voltage if all or the majority of inputs are the same. For example,

```
INPUT_PULLUP = CMOS_LEVEL
```



CFB = Configurable Function Block

```
PIN 12 OUT1 3VOLT
PIN 13 OUT2 5VOLT OPEN_DRAIN
PIN 14 OUT3 5VOLT

OPTIONS DRIVE_LEVEL=3VOLT   ; Changing the default voltage

PIN 13 IN1 5VOLT CMOS_LEVEL  ; Low power inputs
```

Figure 3-3. 3.3V/5V/Open Drain/Low Power Example

Hierarchy

Design hierarchy is supported with the addition of DEFMOD, ENDMOD, and MODULE constructs. Hierarchy with source files allows large designs to be developed from smaller modules or source files. The syntax for calling a lower-level module is,

```
MODULE <module name> [FILE <file name>] ( <argument list> )
```

For <file name>, the .PDS extension should not be used. The <module name> can be the same as the <file name> or can be different. If the <module name> is the same as the <file name>, use only the <file name>, without the .PDS extension.

The syntax used to define a lower-level module is,

```
DEFMOD <module name> ( <argument list> )
ENDMOD
```

where " . . ." refers to valid PDS syntax that defines a function. The parser looks within the same file for module names or in the designated file for filenames. This allows libraries of macros or modules to be developed and used as separate files or as modules within a single file.

3-Level Modular Example

The following is an example that includes three PDS files and three levels of hierarchy.

The first module is a 2-bit XOR design. The module name is **2_bit_xor**, the file name is 2BXOR.PDS. This is the lowest level function for this example.

```
DEFMOD 2_bit_xor (in1, in2, out)    ; referenced by 2BCMPR.PDS
CHIP mod Intel_arch
PIN  in1
PIN  in2
PIN  out

EQUATIONS
    out1 = in1 *  /in1  +  /in1  *  in2
ENDMOD
```

The second module is a 2-bit comparator. The module name is **2_bit_compare**, the file name is 2BCMPR.PDS. This module makes two calls to **2_bit_xor** in 2BXOR.PDS.

```
DEFMOD 2_bit_compare (a1, a2, b1, b2, cmp1, cmp2); referenced by
                                                 ; GATE.PDS
CHIP mod Intel_arch
PIN  a1
PIN  a2
PIN  b1
PIN  b2
PIN  cmp1
PIN  cmp2

MODULE 2_bit_xor FILE 2bxor(in1=a1, in2=b1, out=cmp1)
MODULE 2_bit_xor FILE 2bxor(in1=a2, in2=b2, out=cmp2)
ENDMOD
```

At the highest level for this example, HIGATE1.PDS calls 2_bit_compare in 2BCMPR.PD. Figure 3-4 illustrates the hierarchy for this example.

```
CHIP mydesign iPLD22v10

PIN  addr1
PIN  addr2
PIN  baddr1
PIN  baddr2
PIN  we1
PIN  we2

MODULE 2_bit_compare FILE 2bcmpr (a1=addr1, a2=addr2, b1=baddr1,
                                  b2=baddr2, cmp1=we1, cmp2=we2)
```

HIGATE2.PDS shows how to use the same constructs in a single source file. In this case, the file references itself. This capability is similar to a software program in which a main loop calls subroutines in the same file. (The design Merge utility provides a semi-automated method of using this syntax to create a higher-level PDS file from multiple lower-level PDS files.)

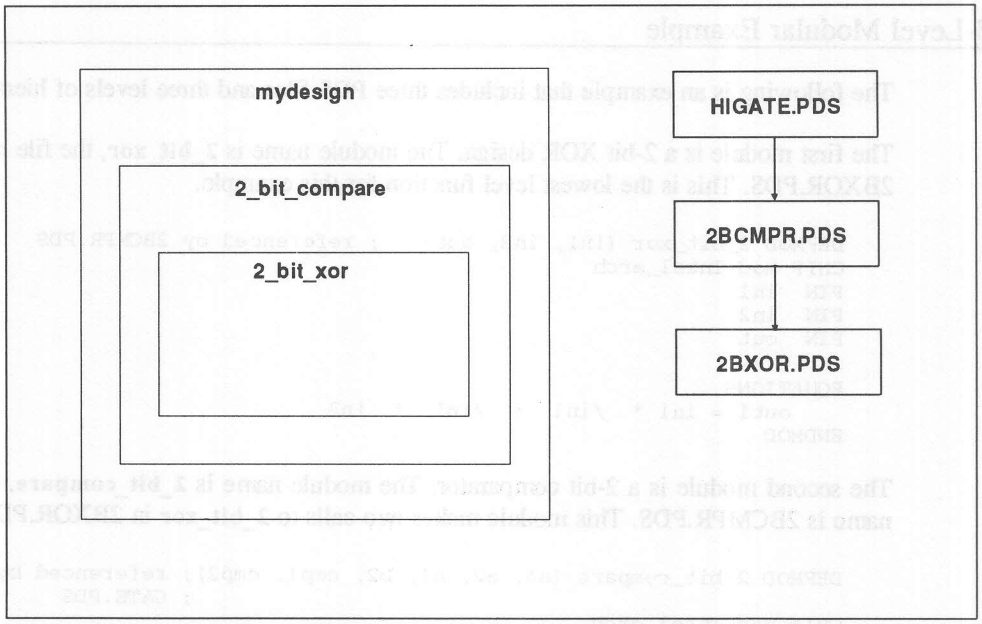


Figure 3-4. Hierarchy Example

NOTE

The ability to independently specify file name and module names allows collections of modules to be grouped into library files.

Merging

The merge feature uses the MODULE reference statement with user-friendly menus to combine PDS designs while preserving user truth tables and state machines.

Vector Notation in Design Section

Vector/set notation is now supported in the design section. For example,

```
OUT[0:3] := A[3:6] * B[0:3]
```

is equivalent to,

```
OUT0 = A3 * B0
OUT1 = A4 * B1
OUT2 = A5 * B2
OUT3 = A6 * B3
```

Groups of signals could be defined in the pin declaration as,

```
PIN   bus[0:31]
```

Or, they could be defined from existing signals as,

```
VECTOR addr := [addr6, addr5, addr4, addr3, addr2, addr1, addr0]
```

Ranges of sequentially numbered signals can be described similarly:

```
x[0:7]
```

is the same as,

```
x0, x1, x2, x3 ,x4, x5, x6, x7
```

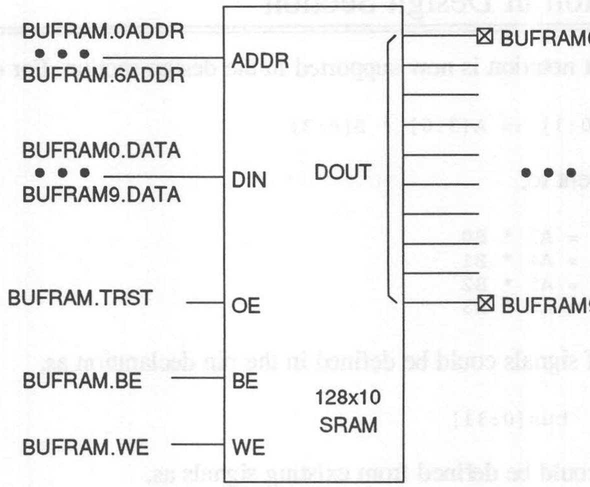
Also, x[ ] means "all signals in the set x." For example, when eight signals x[0:7] are defined in the pin declarations, x[ ] later in the source file is the same as x[0:7]. Note that the order of signals in the pin assignments is assumed through the PDS file unless otherwise specified.

RAM

CFBs in the FPGA device can be configured as a block of 128x10 SRAM instead of a PLD block. An SRAM block may have up to ten outputs, three active-low control inputs (block enable, write enable, and output enable), ten data inputs, and seven address control lines. Figure 3-5 is an example of the PDS specification of a RAM block.

Additionally, individual outputs may be buried (either by the physical device, or by the user), by grounding their OEs. For example, the RAM bits 5 and 7-9 could be buried by,

```
BUFRAM5.TRST = GND
BUFRAM[7:9].TRST = GND
```



Example Syntax:

```
PIN  BUFRAM[0:9]   RAM   BUFRAM
BUFRAM[0:6].ADDR = <addr6, addr5, ..., addr0>
BUFRAM[0:9].DATA = <data9, data8, ..., data0>
BUFRAM.TRST = oe   ; control signals are single inputs
BUFRAM.BE = ce              ; not control terms
BUFRAM.WE=we
```

Figure 3-5. SRAM Example

A RAM block can be initialized with a RAM_DEFAULTS section in the PDS file. This would set JEDEC bits so that at start-up time the memory would be initialized to the specified values. For example,

```
RAM_DEFAULTS          myrom
      ;address        value
          0           0xF
      [1:F]     :     0x0
      [10:7F]   :     0x3FF      ;or DEFAULT_VALUE 0x3FF
```

would set the first 10-bit word to the value of 15, the next 15 words to the value of 0, and the rest of the bits to all 1s.

By tying the Write Enable to GND, the SRAM block becomes a ROM.

The keyword DEFAULT_VALUE followed by a value can be used to initialize the whole SRAM to the same value.

Access to All Internal Signals

In order to access all output/control signals for macrocells and SRAM blocks (to check/view their values during simulation), the following signal name extension forms are accepted in the simulation section:

OUT.SETF	OUT.RSTF
OUT.TRST	OUT.CLKF
OUT.ACLK	OUT.LE
OUT.ALE	OUT.WE
OUT.BE	OUT.ADDR
OUT.DATA	OUT.CMP
OUT.D	OUT.T
OUT.J	OUT.K
OUT.S	OUT.R
OUT.C	OUT.L

Delayed Clock

There are three clocking modes on the FPGA device:

- Synchronous — as with a standard PLD
- Synchronous with delay — provides a slight delay between the input pin and macrocell to shift the t_{SU}, t_{CO} window with respect to the clock.
- Asynchronous — as with a standard PLD that supports asynchronous clocking; this provides yet another shift of t_{SU} and t_{CO} with respect to the clock.

The delayed synchronous mode is supported with the addition of a DELAYCLK pin declaration keyword to be applied to outputs that should use the delayed clock. The syntax is,

```
DELAYCLK
```

There are two synchronous clocks feeding each block in the device. A delayed clock is applied on a block-by-block basis, and thus becomes another grouping criteria (e.g., 3.3VOLT/5VOLT, RAM). For example,

```
PIN  1  clk     ; synchronous clock
PIN     out1    DELAYCLK
PIN     out2    DELAYCLK
PIN     out3
...
out1.CLKF = clk    ;uses delayed clk
out2.CLKF = clk    ;uses delayed clk
out3.CLKF = clk    ;uses standard clk
```

Buried Macrocells

Buried macrocells can be specified with the addition of a NODE keyword in place of a PIN declaration. For example,

```
NODE   234   x_buried
```

is equivalent to,

```
PIN    234   x_buried
...
x_buried.TRST = GND
```

User Electronic Signature (UES) Bits

Support for the user electronic signature bits (UES) is provided by using the SIGNATURE statement. For example,

```
OPTIONS:
   SIGNATURE = 0xffae2
```

The SIGNATURE bits can be read out of the JTAG port using the USERCODE instruction.

New I/O, Feedback Macrocell Combinations

The FPGA device allows for combinatorial/register feedback independent of the output type. This means that combinatorial output is now possible with T, SR, and JK flip-flops. Also, T, SR, and JK feedback is possible with combinatorial output. These architectural combinations are supported with three new pin declaration keywords: TREGFBK, JKFBK, and SRFBK. For example,

```
PIN   tout      CMBFBK
PIN   jkout     CMBFBK
PIN   srout     CMBFBK
PIN   cout_t    TREGFBK
PIN   cout_jk   JKFBK
PIN   cout_sr   SRFBK
tout.T = ...
jkout.J = ...
jkout.K = ...
srout.S = ...
srout.R = ...
cout_t = ...
cout_jk = ...
cout_sr = ...
```

defines,

```
tout.T as T output, combinatorial feedback
jkout as JK output, combinatorial feedback
srout as SR output, combinatorial feedback
cout_t as combinatorial output, T register feedback
cout_jk as combinatorial output, JK flip-flop feedback
cout_sr as combinatorial output, SR flip-flop feedback
```

New Keywords

The following is a list of the new keywords for PLDshell v2.6:

3VOLT	INPUT_PULLUP
5VOLT	JKFBK
CMBFBK	MODULE
CMOS_LEVEL	NODE
DEFAULT_VALUE	OPEN_DRAIN
DEFMOD	RAM
DELAYCLK	RAM_DEFAULTS
DRIVE_LEVEL	SRFBK
ENDMOD	TREGFBK
FILE	TTL_LEVEL
GROUP	

For a list of keywords supported by PLDshell Plus, consult your *PLDshell Plus/PLDasm User's Guide*, V2.1

Chapter 4 — Using Merge

This section uses a merging example to describe using the Merge utility program.

Merge Example

Figure 4-1 shows a block diagram of the example design. The ROM block (dashed lines) are not included in the design. It is present to show a possible additions to the design. The level translator is built into the iFX780 output buffers.

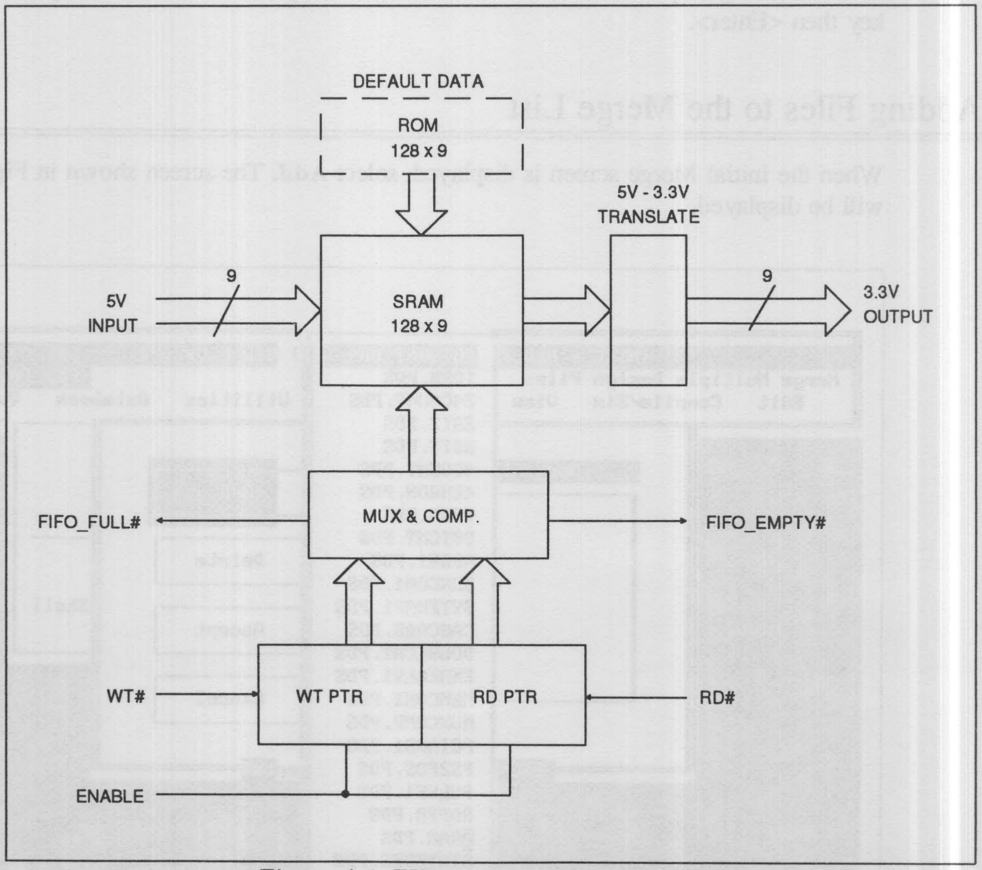


Figure 4-1. FIFO Merge Block Diagram

The merge example design uses four files,

SRAM.PDS — SRAM Block
MUXCOMP.PDS — Multiplexer and Comparator
WTPTR.PDS — Write Pointer
RDPTR.PDS — Read Pointer

which will be merged into a single file, FIFO.PDS. A simulation section can then be added with a text editor (FIFO1.VEC). The design can be estimated and simulated. After estimating, the Estimator Report (FIFO.EST) will be displayed.

The files listed above are located in the installation directory for PLDshell Plus. Also, a file called FIFO1.PDS is in the same directory if you want to skip over the exercise and go right to the results.

Accessing Merge

To select Merge, open the **Utilities Menu**, select Merge with the mouse or press the 'M' key then <Enter>.

Adding Files to the Merge List

When the initial Merge screen is displayed, select **Add**. The screen shown in Figure 4-2 will be displayed.

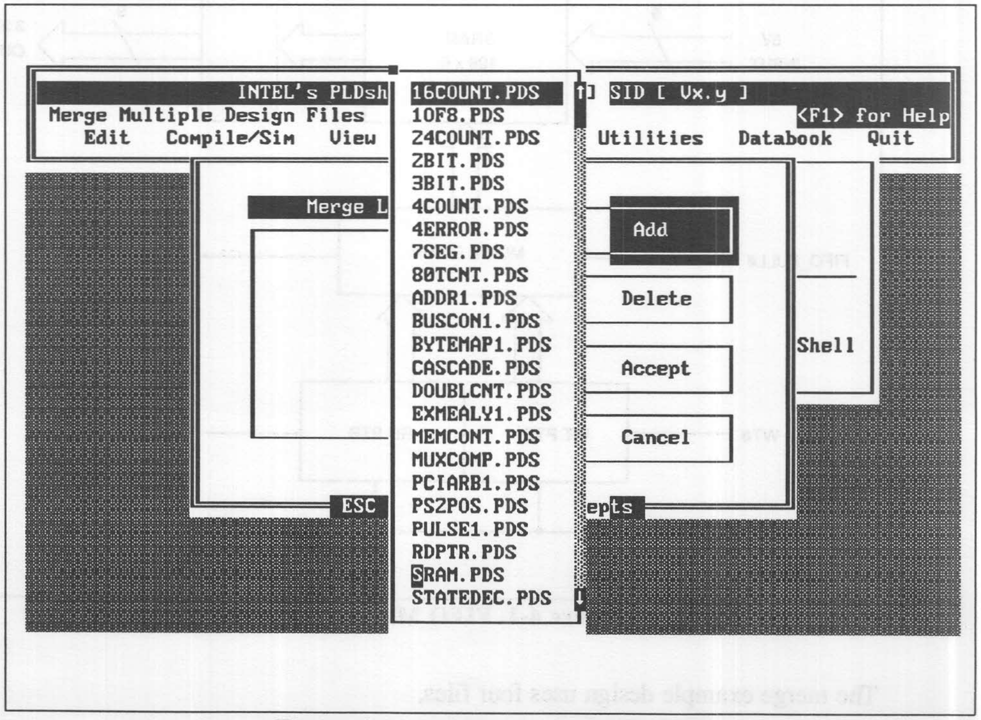


Figure 4-2. Add Screen with File List

Select each of the following four files, one at a time (if you use the sequence indicated, all screens will match the figures exactly; choosing a different order will change the order of signals on the screens, but overall the information will be the same):

SRAM.PDS
MUXCOMP.PDS
WTPTR.PDS
RDPTR.PDS

When all four files have been added to the Merge List, the filenames will appear in that list. Select **Accept**.

NOTE

If you inadvertently select the **Cancel** button, the **Utilities Menu** will appear and all files in the Merge List will be deleted.

Use Original Pin Assignments Screen

Figure 4-3 shows the **Use Original Pin Assignments** screen.

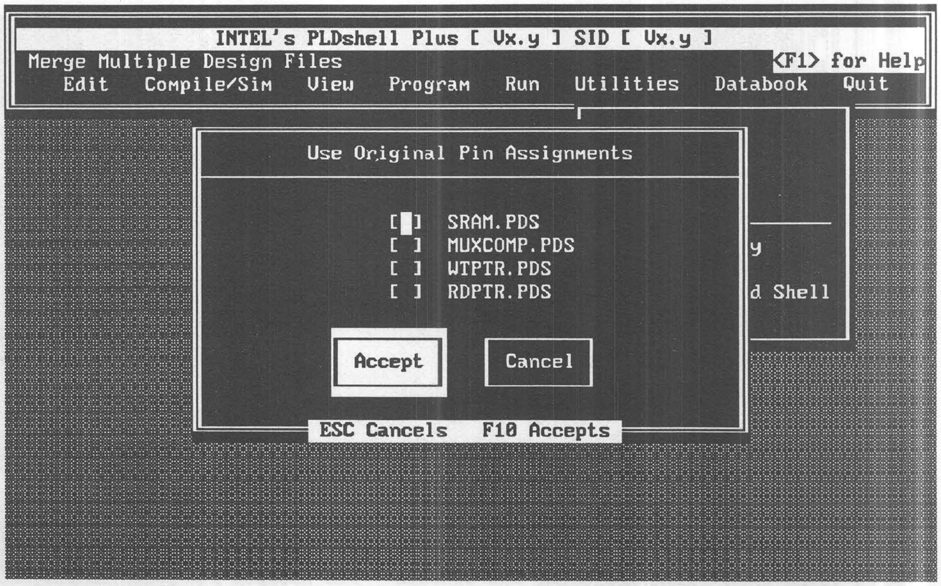


Figure 4-3. Use Original Pin Assignments Screen

All files in the Merge List will be displayed with brackets in front of each filename. An "X" placed in a pair of brackets indicates that the original pin assignments would be used for that particular file. For the purposes of this example, the original pin assignments will *not* be used.

At a later stage of a design, such as when circuit boards have been designed, using the original pin assignments would be desirable to match pin assignments to the actual layout. Generally, when merging two or more designs into a single PDS file in a new design, do not use the original pin assignments.

Select **Accept**. This will display the **Resolve I/O Signals Names/Types** screen.

NOTE

> If you select **Cancel** by mistake, the initial Merge screen will be displayed with the four files listed. You will need to select **Accept** to re-display this screen.

Resolve I/O Signals Names/Types Screen

Figure 4-4 shows the **Resolve I/O Signals Names/Types** screen.

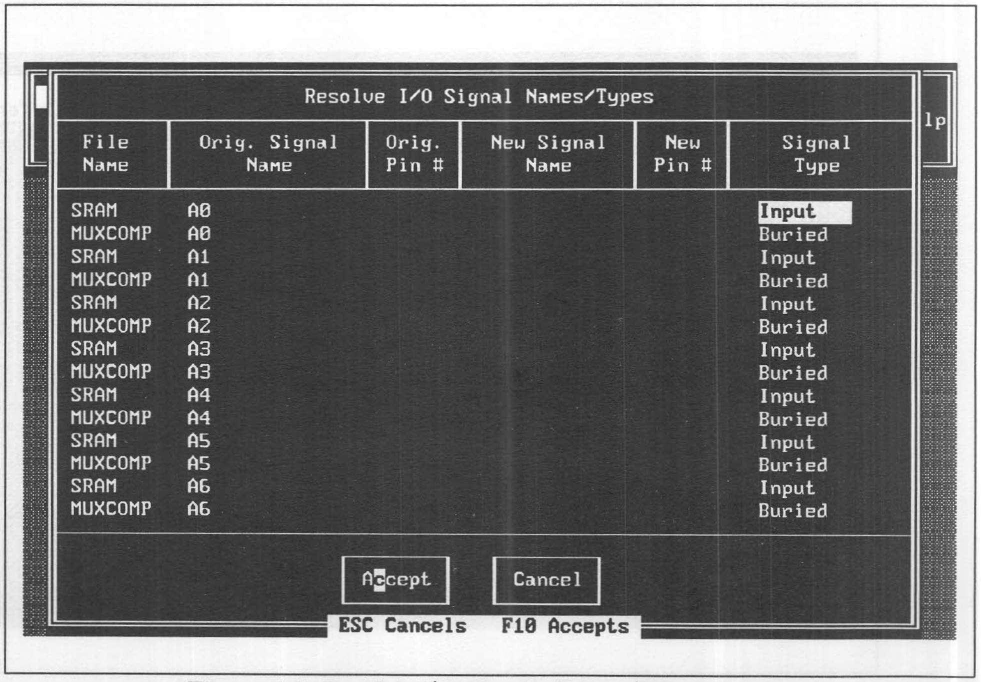


Figure 4-4. Resolve I/O Signals Names/Types Screen

Note that the signal names are in alphabetical order. Where duplicate names exist (because they are used in more than one file, the signal associated with the first file selected is shown first). *Merging assumes that all signals with the same name are to be connected together.*

In this design example, you don't need to edit any of the fields. But to illustrate how to change signals, we'll change all MUXCOMP address outputs (A0 though A6) to "Buried." Click on each "Output" and the label with change to "Buried." Signal names and Pin Assignments are changes by clicking on the respective field and entering text.

Select **Accept**. The **Merge Design Files** screen will be displayed.

NOTE

If you select **Cancel** by mistake, the initial Merge screen will be displayed with the four files listed. You will need to select **Accept** again to re-display this screen.

Merge Design Files Screen

Figure 4-5 shows the **Merge Design Files** screen. This screen contains two fields and the **Accept** and **Cancel** buttons:

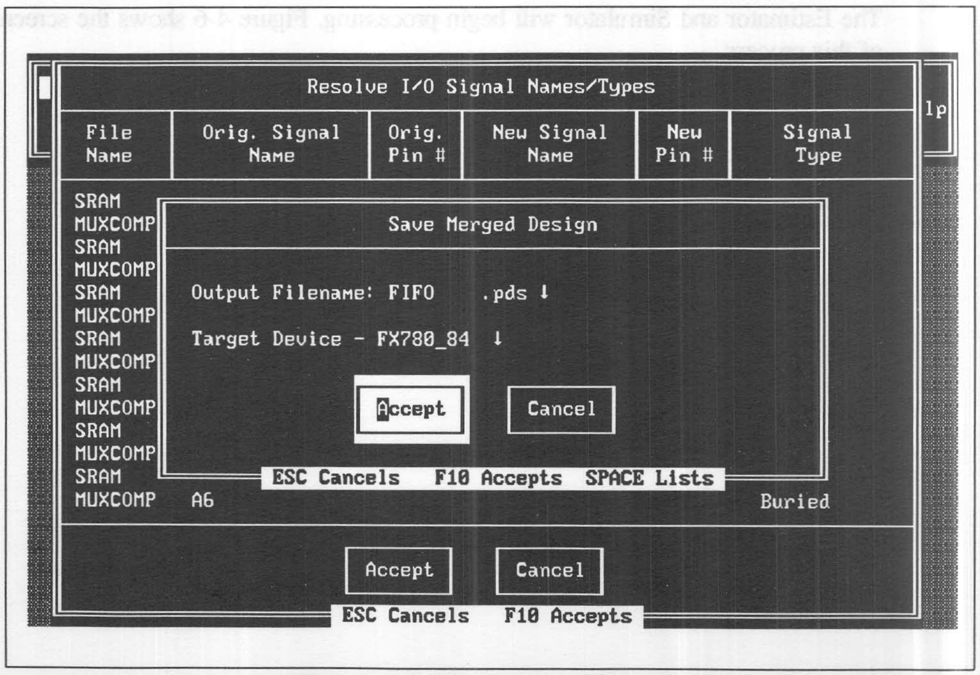


Figure 4-5. Merge Design Screen

In the Output Filename field, type FIFO and press <Enter>.

Double click in the Target Device field and select the FX780_84 device (84-pin version of the iFX780 FPGA).

When the correct Output Filename and Target Device have been entered, select **Accept**.

The message, "Design Merge Successful!" will be displayed. Clicking on the "Okay" button will return you to the **Utilities Menu**.

Estimating and Simulating the Design

At this point, the four files have been merged into FIFO.PDS. Use your text editor to append FIFO1.VEC to the end of FIFO.PDS. Select the **Compile/Sim Menu**. The merged filename, FIFO.PDS will be displayed in the Source Filename field.

Double click in the Processing field. Select Estimate Only

Click on the Compile Options button. Set the Auto Display Report to "Yes." Select the **Accept** button.

The **Compile/Sim Menu** will again be displayed. Select the **Accept** button.

The Estimator and Simulator will begin processing. Figure 4-6 shows the screen display of this process.

```
INFO PARPDS:  Parsing file:  fifo.pds.
INFO PDSTODDB:  Expanding module reference in file:  sram.pds.
INFO PDSTODDB:  Expanding module reference in file:  muxcomp.pds.
INFO PDSTODDB:  Expanding module reference in file:  wtptr.pds.
INFO PDSTODDB:  Expanding module reference in file:  rdptr.pds.
INFO PARPDS:  File parsed correctly.
INFO PARPDS:  Equation expansion complete.
INFO MIN: Espresso II-mv Copyright 1985 U.C. Berkeley Regents
INFO MIN: 66 Equations Minimized
INFO pldpick: Performing estimation analysis on design fifo1.
Estimation analysis completed successfully -
 2 of 10 devices checked may potentially fit design.
SENGN Release [ 2.6 ] SID [ 2.123 ]

INFO SIM: Library [ flipflop.mac ] Loaded!
INFO SIM: Library [ primtive.mac ] Loaded!

INFO SIM: Simulating file: fifo1.sim.
INFO SIM: Vectors Generated       - ( 10 )
INFO SIM: Vectors Generated       - ( 20 )
INFO SIM: Vectors Generated       - ( 30 )
INFO SIM: Vectors Generated       - ( 40 )
INFO SIM: Vectors Generated       - ( 50 )
INFO SIM: Vectors Generated       - ( 60 )
INFO SIM: Vectors Generated       - ( 70 )
INFO SIM: Vectors Generated       - ( 80 )
INFO SIM: Total Vectors Generated - ( 85 )
```

Figure 4-6. Estimator/Simulator Processing Screen

Estimator Report

When processing is complete, press <Enter> to display the Estimator Report. This report consists of two parts: The Estimator Report (Figure 4-7) and a listing of the merged PDS file that was estimated.

To view the simulation waveforms, use the Vector/Waveform Files option of the **View Menu**.

```
DEVICE LISTED IN DESIGN: FX780_84
STATUS:   OK
              ACTIVE                   TOTAL       TOTAL
              PINS        MCELLS       PTERMS      DEVICE
              ---------  ---------  ---------   ---------
FX780_84      39/62      27/80        25%          41%


*** POTENTIAL DEVICES THAT MAY FIT fifo ***

               ACTIVE                   TOTAL       TOTAL
               PINS        MCELLS       PTERMS      DEVICE
              -------------------------------------------------
FX780_84       39/62       27/80        25%          41%
FX780_132      39/104      27/80        25%          35%

*** DEVICES REJECTED FOR fifo ***
   85C220      Not enough input+output pins for this design
   85C224      Not enough input+output pins for this design
  PLD22V10     Not enough input+output pins for this design
  85C22V10     Not enough input+output pins for this design
   PLD610      Not enough input+output pins for this design
   PLD910      Not enough input+output pins for this design
   5AC312      Not enough input+output pins for this design
   5AC324      Not enough input+output pins for this design
```

Figure 4-7. Estimator Report Listing

Chapter 5 — Estimating a Design

Description

The Estimator analyzes a design and provides a list of devices into which a given design may fit. It then reports which devices could potentially implement the design and which devices are rejected because they lack the features needed by the design.

The selection/rejection of devices is based upon high-level, basic criteria such as the number of I/Os, number of macrocells, number of equations, and the availability of certain device-specific such as SRAM, identity comparators, programmable output drivers, and other requirements of a design. No extensive SOP term tests or control term tests are performed.

Estimator Data Inputs/Outputs

The Estimator input data files and output report files are shown in Figure 5-1.

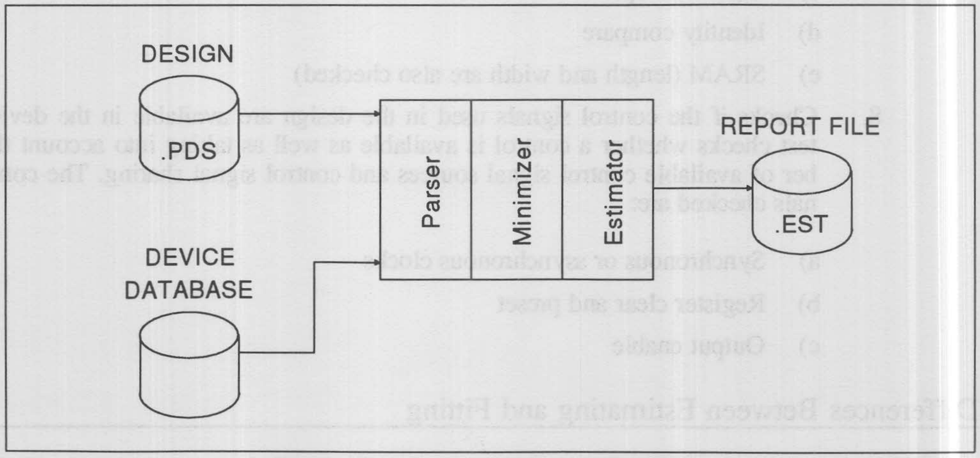


Figure 5-1. Estimator Processing Flow

Estimator Processing

The Estimator performs the following checks to determine if the design will potentially fit into one of the devices in the database:

1. Compares the number of inputs and outputs in the design to the number of active pin device pins.

2. Compares the total number of output pins in the design to the number of device output pins.

3. Compares the number of macrocells in the design to the number of device macrocells.

4. Compares the number of p-terms in the design to the maximum number of device p-terms.

5. Compares the size of p-terms in design to the maximum p-term size supported by the device.

6. Checks if the types of outputs (register types or combinatorial) in the design are supported by the device.

7. Checks if any special device features in the design are supported by the device, such as,

   a) Open drain outputs

   b) 3.3-volt/5-volt outputs

   c) Input Pull-ups

   d) Identity compare

   e) SRAM (length and width are also checked)

8. Checks if the control signals used in the design are available in the device. This test checks whether a control is available as well as taking into account the number of available control signal sources and control signal sharing. The control signals checked are:

   a) Synchronous or asynchronous clocks

   b) Register clear and preset

   c) Output enable

Differences Between Estimating and Fitting

The Estimator takes a high-level look at the overall utilization of device resources which are consumed by a design. The fitting process handles checking and resolving the more complex issues of signal routing, specific sizes of equations for control signals, the allocation of p-terms to macrocells, and other considerations.

Interpreting Estimator Results

After estimation is completed, the devices which have passed the tests described earlier are listed as being potential candidates for a fit in the Estimation Report File (.EST). The list reports utilization percentages for the key resources consumed by the design and a “Total Device Utilization” percentage. A list of rejected devices is also generated which includes the first reason for failure.

There is some probability the design will not fit into the device(s) which pass the Estimator's tests, due to issues that can only be resolved by the fitter. The goal of the Estimator is to provide a quick method of narrowing the choice of a target device for a design implementation. The Estimator also gives the designer some idea of how much of a device's resources will be utilized by a given design. This information, coupled with general guidelines on the ability of a device's architecture to accommodate designs at various levels of resource utilization, can be a helpful indicator of which device to target for the final implementation.

It may be helpful to think of the Estimator as a filter which eliminates devices that lack the necessary resources from consideration. This helps in making the choice of the most effective device from a device utilization standpoint.

Estimator Report File

The Estimator Report File is shown in Figure 5-2. The sections commented with a "#" are described in the following paragraphs.

"Device does not have set (.SETF) feature"
Design specified register preset control not supported on device.

"Design has too many Output/Macrocell clocks for device to support"
Design specified more register clock signal sources than the device supports.

"Asynchronous Array clocking not available"
Design specified asynchronous clock not supported on device.

"Identity comparator (.CMP) not available"
Design specified identity compare feature not supported by device.

"Device does not support output of type '<type>'"
Design specified a register or output type not supported by device.

"Not enough p-terms on device for this design"
Design requires more p-terms than the device has available.

"Device does not have large enough SOP"
Design specified a product term too large to implement on the device.

"Not enough macrocells on device"
Design requires more macrocells than are available on the device.

"Not enough I/O pins on device"
Design requires more I/O pins than are available on the device.

"Not enough inputs+output pins on device"
Design requires more pins than are available on the device.

```
PLDpick [ R2.x ] SID [ x.y ]
DEVICE ESTIMATION REPORT FOR FIFO1

*** DESIGN STATISTICS ***
    Inputs              5  # number of inputs in design  #
    Outputs             8  # number of outputs in design #
    Active Pins (in+out)    13  # number of active pins in design #

           Combinatorial      0   # number of combinatorial macrocells  #
           D flip-flops      15   # number of macrocells using D flip-flops #
           T flip-flops       1   # number of macrocells using T flip-flops #
         J/K (emulation)      0   # number of macrocells using JK flip-flops #
    Total Macrocells         16        Buried Macrocells       8
    # buried macrocells use feedback only #
    Largest Product Term    14 # size of largest product term in design #

    DEVICE LISTED IN DESIGN: 5AC324 # part specified by design #
    STATUS:   OK  # the part specified in the design passes the checks  #
 # next two sections summarize how the resources of the parts checked     #
 # would be consumed by the design                                        #
             ACTIVE                  TOTAL       TOTAL
             PINS        MCELLS     PTERMS      DEVICE
             -----       ------     ------      ------
     5AC324   13/36        16/24       2%          41%

*** POTENTIAL DEVICES THAT MAY FIT design ***

             ACTIVE                  TOTAL        TOTAL
             PINS        MCELLS     PTERMS       DEVICE
             -----       ------     ------       ------
      PLD910    21/34      16/24      4%           42%
      5AC324    13/36      16/24      2%           41%
    FPGA8F68    13/50      16/80      0%           18%
    FPGA8F84    13/62      16/80      0%           16%
   FPGA8F100    13/76      16/80      0%           14%
   FPGA8F132   13/104      16/80      0%           13%

# this lists all devices rejected for implementing the design #
# and the reason for rejection                                #

*** DEVICES REJECTED FOR design ***
    85C220          Not enough input+output pins on device
    85C224          Not enough I/O pins on device
   PLD22V10         Not enough I/O pins on device
    PLD610          Not enough input+output pins on device
    5AC312          Not enough macrocells on device
```

Figure 5-2. Estimator Report File

"Device does not have on-board SRAMS"
Design specified a RAM block which is not available on the device.

"Not enough SRAMs on device for design"
Design specified more RAM blocks than are available on the device.

"SRAMs on device not long enough: need <num> depth, max. <num> available"
Design specified a RAM block larger than available on the device.

"Device does not have latched inputs"
"Device does not have any 3.3 volt outputs"
"Device does not have any Open-Drain outputs"
Design specified use of a feature not supported by the device.

Estimator Failure Codes and Messages

EST-I6823 "Generating Estimation Analysis of <design>"
Message indicating design process started.

PICK-I6824 "Estimation Analysis Completed Successfully."
Message indicating processing completed without errors.

PICK-E6801 "Unable to open file for reading: <file>"
The <file> could not be found and/or opened.

PICK-E6802 "Unable to open file for writing: <file>"
The <file> could not be found and/or opened.

PICK-E6803 "Part Database possibly corrupted. Unknown db key: <key>"
The device database contains an unrecognized keyword <key>.

PICK-E6804 "Wrong Part Database. Version is incorrect"
The device database is not the correct version for use with the current version of the Estimator, or it has an incompatible database.

Chapter 6 — Design Examples

This section provides design examples using iFX780 devices. The PDS files are installed with PLDshell Plus from the distribution diskettes.

Sample Designs

Table 6-1 lists the filenames and descriptions of the sample designs.

Table 6-1. Design Examples

Filename	Description
BYTEMAP1.PDS	An example design for mapping 32-bit CPU data in 8-bit blocks using the iFX780.
SRAM.PDS MUXCOMP.PDS WTPTR.PDS RDPTR.PDS FIFO1.VEC	Four files that can be merged into a 128 x 9-bit wide FIFO. RDPTR = Read Pointer, WTPTR = Write Pointer, MUXCOMP = Multiplexer/Comparator, SRAM = SRAM Definition. FIFO1.VEC is a simulation vector file that can be appended to the merged design for simulation. FIFO1.PDS is also installed to show the final merged design.
3BIT.PDS 10F8.PDS WIGGLE.VEC	Two files that can be merged into a simple pattern generator. 3BIT = 3-bit state machine, 1OF8 = 1 of 8 decoder. WIGGLE.VEC is a simulation vector file that can be appended to the merged design for simulation. WIGGLE.PDS is also installed to show the final merged design.
80TCNT.PDS	80-bit counter using Toggle flip-flops.
PCIARB1.PDS	PCI bus arbiter design using a fixed-priority arbitration and supporting 10 masters.
*2BXOR.PDS *2BCMPR.PDS *HIGATE1.PDS *HIGATE2.PDS	Example files for illustrating modular design syntax. HIGATE1.PDS calls 2BCMPR.PDS, which in turn calls 2BXOR.PDS. HIGATE2.PDS combines the same functionality into a single source file.
*PATGEN1.PDS	Serial pattern generator that stores a default pattern in SRAM. Consecutively reads 10 bits of data from SRAM, serializes it, and shifts it out.

*These files are not installed automatically. They must be copied from Disk #2.